import streamlit as st

from pymongo import MongoClient

from langchain\_openai import ChatOpenAI

from langchain.prompts import PromptTemplate

from langchain.chains import LLMChain

import re

from pymongo import MongoClient

import gridfs

import base64

import io

from datetime import datetime, time

# Simulated user database

client = MongoClient('mongodb+srv://krrish852456:krrish852456@cluster0.99khz.mongodb.net/?retryWrites=true&w=majority&appid=Cluster0')

db = client["slms"]

dba=client['assign']

collection = db["slms"]

collection1 = db["Subjects"]

collection2 = db["Instructor"]

cust = db["Customer Care"]

assign=db["assign"]

sum=db["Summary"]

m=db["modules"]

p=db["payment"]

fs = gridfs.GridFS(db)

fsa = gridfs.GridFS(dba)

# Session state initialization

if "reg\_in" not in st.session\_state:

st.session\_state["reg\_in"] = False

if "logged\_in" not in st.session\_state:

st.session\_state["logged\_in"] = False

st.session\_state["userid"] = ""

st.session\_state["role"] = ""

if "messages" not in st.session\_state:

st.session\_state.messages = {} # Store chat history per user

if "admin\_joined" not in st.session\_state:

st.session\_state.admin\_joined = {}

#Logout Function

def logout():

st.session\_state["logged\_in"] = False

st.session\_state["userid"] = ""

st.session\_state["role"] = ""

st.session\_state["rerun"] = True

st.rerun()

**User Management**

* + Student, Instructor, and Admin account creation & management
  + Role-based access control

#Reg function

def reg():

'Already have an Account'

if st.button('Login Page'):

st.session\_state["reg\_in"] = False

st.session\_state["rerun"] = True

st.rerun()

new\_userid = st.text\_input("New Userid")

new\_password = st.text\_input("New Password", type="password")

s=st.text\_input("Specialization")

if st.button("Register"):

if collection.find\_one({"id": new\_userid}) is not None:

st.warning("The UserID already EXIST.")

else:

if new\_userid and new\_password:

y={"id":new\_userid,"pwd":new\_password,"role":'Student',"spec":s,"bal":300}

collection.insert\_one(y)

st.success('Registered Sucessfully')

else:

st.warning("Please enter both userid and password.")

st.session\_state["reg\_in"] = False

st.session\_state["rerun"] = True

st.rerun()

**Content Management**

* + Upload and organize multimedia content (videos, PDFs, quizzes)
  + Provide downloadable course materials

def display\_pdf(pdf\_bytes):

base64\_pdf = base64.b64encode(pdf\_bytes).decode('utf-8')

pdf\_display = f'<iframe src="data:application/pdf;base64,{base64\_pdf}" width="700" height="600"></iframe>'

st.markdown(pdf\_display, unsafe\_allow\_html=True)

#binary\_data = pdf\_file.getvalue()

#pdf\_viewer(input=binary\_data,width=700)

**Content Management**

* + Upload and organize multimedia content (videos, PDFs, quizzes)
  + Provide downloadable course materials

def retrival(c,i,o):

module=[]

pdf\_files = list(db.fs.files.find({}, {"metadata": 1}))

if pdf\_files!=[]:

for file in pdf\_files:

metadata = file.get("metadata", {})

print(metadata)

if metadata['course']==c and metadata['id']==i and metadata['option']==o:

module.append(metadata['name'])

n=m.find({'id':i,'option':o},{"name":1})

if n is not None:

for f in n:

module.append(f['name'])

selected\_filename = st.selectbox("Select a PDF to View",module)

query = {"metadata.name": selected\_filename}

results = list(db.fs.files.find(query, {"metadata": 1}))

print('results')

print(results)

if results !=[]:

st.write(results[0]['metadata']['description'])

file\_id = results[0]['\_id']

print('p')

print(file\_id)

pdf\_data = fs.get(file\_id)

if results[0]['metadata']['filename'].split(".")[1]=='pdf':

display\_pdf(pdf\_data.read())

st.download\_button(label="Download PDF", data=pdf\_data.read(), file\_name=selected\_filename)

n=m.find\_one({"name":selected\_filename})

if n is not None:

st.write(n['description'])

return(selected\_filename)

**Grading System**

* + Auto-grade MCQs
  + Manual grading for descriptive answers with feedback

def retrivala(c,i):

module=[]

pdf\_files = list(db.fs.files.find({}, {"metadata": 1}))

if pdf\_files!=[]:

for file in pdf\_files:

metadata = file.get("metadata", {})

if metadata['course']==c and metadata['id']==i and metadata['option']=='Assesment':

module.append(metadata['name'])

n=m.find({'id':i,'option':'Assesment'},{"name":1})

if n is not None:

for f in n:

module.append(f['name'])

selected\_filename = st.selectbox("Select a PDF to View",module)

query = {"metadata.name": selected\_filename}

results = list(db.fs.files.find(query, {"metadata": 1}))

if results!=[]:

# Dropdown to select a PDF file

# Retrieve the PDF file from MongoDB

file\_id = results[0]['\_id']

pdf\_data = fs.get(file\_id).read()

if results[0]['filename'].split(".")[1]=='pdf':

display\_pdf(pdf\_data.read())

# Convert to bytes and display

st.download\_button(label="Download PDF", data=pdf\_data, file\_name=selected\_filename)

print('r')

print(results)

if results[0]['choice']=='Descriptive':

st.write(results[0]['metadata']['description'])

ans=st.text\_area('Enter the Answer')

**Grading System**

* + Auto-grade MCQs
  + Manual grading for descriptive answers with feedback

if results[0]['choice']=='MCQ':

ans=st.radio(results[0]['metadata']['description'],[results[0]['metadata']['a']])

if results[0]['choice']=='More than One Answer MCQ':

ans=[]

for i in results[0]['metadata']['a']:

ci=st.checkbox(i)

ans.append(ci)

if results[0]['choice']=='True or False':

ans=st.radio(results[0]['metadata']['description'],[True,False])

assign.insert\_one({'course':c,'ins':i,'mod':selected\_filename,'ans':ans})

n=m.find\_one({"name":selected\_filename})

if n is not None:

if n['option']=='Assesment':

if n['choice']=='Descriptive':

st.write(n['description'])

ans=st.text\_area('Enter the Answer')

if n['choice']=='MCQ':

ans=st.radio(n['description'],n['a'])

if n['choice']=='More than One Answer MCQ':

ans=[]

for i in n['a']:

ci=st.checkbox(i)

ans.append(ci)

if n['choice']=='True or False':

ans=st.radio(n['description'],['True','False'])

if st.button('submit'):

if ans==n['ans']:

st.write('✅Correct')

else:

st.write('Wrong')

**User Management**

* + Student, Instructor, and Admin account creation & management
  + Role-based access control

# Login function

def login():

st.title("Login Page")

userid = st.text\_input("Userid")

password = st.text\_input("Password", type="password")

'Create an Account'

if st.button('Registration form'):

st.session\_state["reg\_in"] = True

st.session\_state["rerun"] = True

st.rerun()

if st.button("Login"):

user = collection.find\_one({"id": userid})

if user is not None:

if user["pwd"] == password:

st.session\_state["logged\_in"] = True

st.session\_state["userid"] = userid

st.session\_state["role"] = user["role"]

st.session\_state["rerun"] = True

st.rerun() # Refresh to show navigation

else:

st.error("Invalid Password")

else:

st.error("Invalid Userid")

**Content Management**

* + Upload and organize multimedia content (videos, PDFs, quizzes)
  + Provide downloadable course materials

def moduleview():

st.title("📄 Retrieve and Display PDFs from MongoDB")

# Fetch all stored PDF files

pdf\_files = list(db.fs.files.find({}, {"filename": 1, "\_id": 1}))

# Dropdown to select a PDF file

if pdf\_files!=[]:

file\_options = {file["filename"]: file["\_id"] for file in pdf\_files}

selected\_filename = st.selectbox("Select a PDF to View", list(file\_options.keys()))

if st.button("Load PDF"):

# Retrieve the PDF file from MongoDB

file\_id = file\_options[selected\_filename]

pdf\_data = fs.get(file\_id).read()

# Convert to bytes and display

st.download\_button(label="Download PDF", data=pdf\_data, file\_name=selected\_filename, mime="application/pdf")

st.write("📄 \*\*PDF Preview:\*\*")

st.pdf(io.BytesIO(pdf\_data))

else:

st.write("⚠️ No PDFs found in the database.")

**Customer Care**

LLM-based customer service for common queries

def mainc():

llm=ChatOpenAI(api\_key='sk-proj-...', #st.secrets["OPEN\_API\_KEY"]

model\_name='gpt-4o',

temperature=0.0)

prompt\_template='''If any actionable prompt is given the state yes else give the response.

Text:

{context}'''

PROMPT = PromptTemplate(

template=prompt\_template, input\_variables=["context"])

i=0

if st.session\_state['userid'] not in st.session\_state['messages']:

st.session\_state['messages'][st.session\_state['userid']] = [{"role": "assistant", "content": "Hello! Welcome to customer care. How Can I help you?"}]

st.session\_state.admin\_joined[st.session\_state['userid']] = False

i=1

if i==1:

st.session\_state["messages"].update([i for i in sum.find({})][0])

i=0

else:

st.session\_state["messages"]=[i for i in sum.find({})][0]

for msg in st.session\_state.messages[st.session\_state['userid']]:

print(msg)

st.chat\_message(msg["role"]).write(msg["content"])

if prompt := st.chat\_input():

st.session\_state["messages"][st.session\_state['userid']].append({"role": "user", "content": prompt})

st.chat\_message("user").write(prompt)

chain = LLMChain(llm=llm, prompt=PROMPT)

answer=chain.run(prompt)

if re.search(r'\bYes\b', answer):

cust.insert\_one({'id':st.session\_state["userid"],'query':prompt})

st.chat\_message("assistant").write("Notified to the Admin, He will get back to you soon...")

st.session\_state.admin\_joined[st.session\_state['userid']] = True

elif not st.session\_state.admin\_joined[st.session\_state['userid']]:

prompt\_template='''Accept the queries as a customer care and give an accuarte reply.

Text:

{context}'''

PROMPT = PromptTemplate(

template=prompt\_template, input\_variables=["context"])

chain = LLMChain(llm=llm, prompt=PROMPT).run(prompt)

st.session\_state["messages"][st.session\_state['userid']].append({"role": "assistant", "content": chain})

st.chat\_message("assistant").write(chain)

i=1

sum.delete\_many({})

sum.insert\_one(st.session\_state["messages"])

# Main app interface Student

def maini():

st.sidebar.title("Navigation")

page = st.sidebar.radio("Go to", ["Dashboard", "Module", "view Assignments","Payment","Customer Care"])

if page == "Dashboard":

st.title("Dashboard")

st.write(f"Hello, {st.session\_state['userid']}! You are logged in as {st.session\_state['role']}.")

documents=collection2.find({"Instructor": {"$in": [st.session\_state["userid"]]}},{"course": 1, "\_id": 0})

key\_values = [doc['course'] for doc in documents if 'course' in doc]

optionm = st.selectbox("Course",(key\_values))

**Assignment System**

* + Instructors create assignments
  + Students submit assignments online

elif page == "Module":

st.title("Module")

st.write("Welcome to the Module creation page.")

flag = st.toggle("AI Correction")

documents=collection2.find({"Instructor": {"$in": [st.session\_state["userid"]]}},{"course": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['course'] for doc in documents if 'course' in doc]

optionm = st.selectbox("Course",(key\_values))

name=st.text\_input("Enter the module name")

e=None

existing\_file = db.fs.files.find\_one({"metadata.name": name})

e=m.find\_one({"name":name})

if existing\_file is not None or e is not None:

st.warning("⚠️ A file with this unique ID already exists! Please enter a different ID.")

else:

description=st.text\_area("Enter the text")

# File uploader widget

uploaded\_file = st.file\_uploader("Upload a PDF file", type=[])

option = st.selectbox("Module",('Learning Content','Assignment','Assesment'))

if option=='Learning Content' or option=='Assignment':

if option=='Assignment':

'Enter Assignment Deadline'

d=st.date\_input("📅Date:",format='MM/DD/YYYY')

t=st.time\_input("⏰Time:",value=time(0,0))

dt= datetime.combine(d,t)

if st.button("Upload"):

if uploaded\_file is not None:

st.success(f"✅ Uploaded: {uploaded\_file.name}")

# Convert file to binary for MongoDB storage

file\_data = uploaded\_file.read()

# Check if file already exists in MongoDB

existing\_file = db.fs.files.find\_one({"filename": uploaded\_file.name})

if existing\_file:

st.warning("⚠️ File already exists in MongoDB.")

else:

# Store file in GridFS

if option=='Learning Content':

file\_id = fs.put(file\_data, filename=uploaded\_file.name,metadata={"filename":uploaded\_file.name,"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option})

st.success(f"📁 File saved to MongoDB with ID: {file\_id}")

else:

file\_id = fs.put(file\_data, filename=uploaded\_file.name,metadata={"filename":uploaded\_file.name,"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'dead':dt})

st.success(f"📁 File saved to MongoDB with ID: {file\_id}")

else:

if option=='Learning Content':

m.insert\_one({"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option})

st.success(f"✅ Uploaded")

else:

m.insert\_one({"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'dead':dt})

st.success(f"✅ Uploaded")

**Assessment**

* + Basic quiz functionality (MCQs & True/False)
  + Auto-grading for MCQs

if option=='Assesment':

m.delete\_many({option:"Assesment"})

choice = st.selectbox("Type of Question",('Descriptive','MCQ','More than One Answer MCQ','True or False'))

if choice=='Descriptive':

if st.button("Upload"):

if uploaded\_file is not None:

st.success(f"✅ Uploaded: {uploaded\_file.name}")

# Convert file to binary for MongoDB storage

file\_data = uploaded\_file.read()

# Check if file already exists in MongoDB

existing\_file = db.fs.files.find\_one({"filename": uploaded\_file.name})

if existing\_file:

st.warning("⚠️ File already exists in MongoDB.")

else:

# Store file in GridFS

file\_id = fs.put(file\_data, filename=uploaded\_file.name,metadata={"filename":uploaded\_file.name,"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'choice':choice})

st.success(f"📁 File saved to MongoDB with ID: {file\_id}")

else:

m.insert\_one({"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'choice':choice})

st.success(f"✅ Uploaded")

if choice=='MCQ' or choice=='More than One Answer MCQ':

n=int(st.text\_input("No. of options:",4))

a=[]

for i in range(n):

x=st.text\_input(f"{i}.")

a.append([x])

if choice=='MCQ':

an=st.text\_input('Ans:')

if choice=='More than One Answer MCQ':

an=[]

n=int(st.text\_input("No. of options:",4))

for i in range(n):

x=st.text\_input(f"{i}.")

an.append([x])

if st.button("Upload"):

if uploaded\_file is not None:

st.success(f"✅ Uploaded: {uploaded\_file.name}")

# Convert file to binary for MongoDB storage

file\_data = uploaded\_file.read()

# Check if file already exists in MongoDB

existing\_file = db.fs.files.find\_one({"filename": uploaded\_file.name})

if existing\_file:

st.warning("⚠️ File already exists in MongoDB.")

else:

# Store file in GridFS

file\_id = fs.put(file\_data, filename=uploaded\_file.name,metadata={"filename":uploaded\_file.name,"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'choice':choice,'a':a,'ans':an})

st.success(f"📁 File saved to MongoDB with ID: {file\_id}")

else:

m.insert\_one({"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'choice':choice,'a':a,'ans':an})

st.success(f"✅ Uploaded")

if choice=='True or False':

an=st.text\_input('Ans:')

if st.button("Upload"):

if uploaded\_file is not None:

st.success(f"✅ Uploaded: {uploaded\_file.name}")

# Convert file to binary for MongoDB storage

file\_data = uploaded\_file.read()

# Check if file already exists in MongoDB

existing\_file = db.fs.files.find\_one({"filename": uploaded\_file.name})

if existing\_file:

st.warning("⚠️ File already exists in MongoDB.")

else:

# Store file in GridFS

file\_id = fs.put(file\_data, filename=uploaded\_file.name,metadata={"filename":uploaded\_file.name,"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'choice':choice,'ans':an})

st.success(f"📁 File saved to MongoDB with ID: {file\_id}")

else:

m.insert\_one({"name": name, "course": optionm, "description": description,'id':st.session\_state['userid'],'option':option,'flag':flag,'choice':choice,'ans':an})

st.success(f"✅ Uploaded")

elif page == "view Assignments":

st.title("view Assignments")

module=[]

documents=collection2.find({"Instructor": {"$in": [st.session\_state["userid"]]}},{"course": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['course'] for doc in documents if 'course' in doc]

optionm = st.selectbox("Course",(key\_values))

pdf\_files = list(db.fs.files.find({}, {"metadata": 1}))

if pdf\_files!=[]:

for file in pdf\_files:

metadata = file.get("metadata", {})

if metadata['course']==optionm:

module.append(metadata['name'])

n=m.find({},{"name":1})

if n is not None:

for f in n:

module.append(f['name'])

selected\_filename = st.selectbox("Select module",module)

pdf\_files = dba.fs.files.find({'filename': {"$regex": selected\_filename}}, {"filename": 1})

print('pdf\_files')

print(pdf\_files)

a=[]

if pdf\_files is not None:

for i in pdf\_files:

print(i)

a.append(i['filename'])

selected\_filename = st.selectbox("Select a PDF to View",a)

file\_id = dba.fs.files.find\_one({"filename":selected\_filename}, {"\_id": 1})

print('id')

print(file\_id)

if file\_id is not None:

pdf\_data = fsa.get(file\_id['\_id']).read()

if 'pdf' in selected\_filename.split('.'):

display\_pdf(pdf\_data)

# Convert to bytes and display

st.download\_button(label="Download PDF", data=pdf\_data, file\_name=selected\_filename)

**Customer Care**

LLM-based customer service for common queries

elif page == "Customer Care":

st.title("Customer Care")

mainc()

if st.button("Logout"):

logout()

# Main app interface Instructor

def mains():

st.sidebar.title("Navigation")

page = st.sidebar.radio("Go to", ["Dashboard", "Module", "Assignment","Assesment","Payment","Customer Care"])

if page == "Dashboard":

st.title("Home Page")

st.write(f"Hello, {st.session\_state['userid']}! You are logged in as {st.session\_state['role']}.")

documents = p.find({'id':st.session\_state['userid']}, {"spec": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

s = st.selectbox("Specialization",(key\_values))

documents = p.find({'id':st.session\_state['userid'],"spec":s}, {"course": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['course'] for doc in documents if 'course' in doc]

c = st.selectbox("course",key\_values)

elif page == "Module":

st.title("Admin Dashboard")

st.write("Welcome to the admin page.")

documents = p.find({'id':st.session\_state['userid']}, {"spec": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

s = st.selectbox("Specialization",(key\_values))

documents = p.find({"spec":s,'id':st.session\_state['userid']}, {"course": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['course'] for doc in documents if 'course' in doc]

c = st.selectbox("course",key\_values)

ins = p.find\_one({"spec":s,'course':c}, {"instructor": 1, "\_id": 0})

if ins is not None:

i=ins['instructor']

retrival(c,i,'Learning Content')

elif page == "Assignment":

st.title("User Dashboard")

st.write("Welcome to the user page.")

documents = p.find({'id':st.session\_state['userid']}, {"spec": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

s = st.selectbox("Specialization",(key\_values))

documents = p.find({"spec":s,'id':st.session\_state['userid']}, {"course": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['course'] for doc in documents if 'course' in doc]

c = st.selectbox("course",key\_values)

ins = p.find\_one({"spec":s,'course':c}, {"instructor": 1, "\_id": 0})

if ins is not None:

i=ins['instructor']

m=retrival(c,i,page)

uploaded\_file = st.file\_uploader("Upload a PDF file", type=[])

if st.button('upload'):

if uploaded\_file is not None:

file\_data = uploaded\_file.read()

display\_pdf(file\_data)

file\_id = fsa.put(file\_data, filename=f'{m}.{i}.{uploaded\_file.name}.{st.session\_state["userid"]}',metadata={'upload\_time':datetime.utcnow()})

st.success(f"📁 File saved to MongoDB with ID: {file\_id}")

elif page == "Assesment":

st.title("Assesment")

st.write("Welcome to the user page.")

documents = p.find({}, {"spec": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

s = st.selectbox("Specialization",(key\_values))

documents = p.find({'id':st.session\_state['userid'],"spec":s}, {"course": 1, "\_id": 0})

print('documents')

print(documents)

if documents is not None:

key\_values = [doc['course'] for doc in documents if 'course' in doc]

if key\_values != []:

c = st.selectbox("course",key\_values)

ins = p.find\_one({"id":st.session\_state['userid'],"spec":s,'course':c}, {"instructor": 1, "\_id": 0})

if ins is not None:

i=ins['instructor']

retrivala(c,i)

elif page == "Payment":

st.title("Payment")

documents = collection1.find({}, {"spec": 1, "\_id": 0})

if documents is not None:

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

s = st.selectbox("Specialization",(key\_values))

if documents is not None:

documents = collection1.find({"spec":s}, {"course": 1, "\_id": 0})

key\_values = [doc['course'] for doc in documents if 'course' in doc]

c = st.selectbox("course",key\_values[0])

exist=p.find\_one({'course':c})

if exist is None:

documents = collection2.find({"spec":s,"course":c}, {"Instructor": 1, "\_id": 0})

key\_values = [doc['Instructor'] for doc in documents if 'Instructor' in doc]

print('key\_values')

print(key\_values)

if key\_values != []:

ins = st.selectbox("Instructor",key\_values[0])

if st.button('pay'):

b=collection.find\_one({'id':st.session\_state['userid']},{"bal":1})

if b['bal']!=0:

i=b['bal']-100

collection.update\_one({"id": st.session\_state['userid']}, {"$set":{'bal':i}})

p.insert\_one({"id":st.session\_state['userid'],"spec":s,"course":c,"instructor":ins})

st.success(f"payed successfully, you have {i} balance in your account")

else:

st.error("No enough balance")

else:

st.warning("Already Payed")

elif page == "Customer Care":

st.title("Customer Care")

mainc()

if st.button("Logout"):

logout()

# Main app interface Admin

def maina():

st.sidebar.title("Navigation")

page = st.sidebar.radio("Go to", ["Home Page","Instruct Reg", "Course Reg", "Course View","Course Assign","Assign View","Notification"])

if page == "Home Page":

st.title("Home Page")

st.write(f"Hello, {st.session\_state['userid']}! You are logged in as {st.session\_state['role']}.")

elif page == "Instruct Reg":

st.title("Instructor Registration")

st.write(f"Hello, {st.session\_state['userid']}! You are logged in as {st.session\_state['role']}.")

new\_userid = st.text\_input("New Userid")

new\_password = st.text\_input("New Password", type="password")

s=st.text\_input("Specialization")

if st.button("Register"):

if collection.find\_one({"id": new\_userid}) is not None:

st.warning("The UserID already EXIST.")

else:

if new\_userid and new\_password:

y={"id":new\_userid,"pwd":new\_password,"role":'Instructor',"spec":s,"bal":300}

collection.insert\_one(y)

st.success('Registered Sucessfully')

else:

st.warning("Please enter both userid and password.")

elif page == "Course Reg":

st.title("Course Registration")

st.write(f"Hello, {st.session\_state['userid']}! You are logged in as {st.session\_state['role']}.")

s=st.text\_input("Specialization")

c=st.text\_input("Course")

exist=list(collection1.find({'spec':s,'course':c}))

print('exist')

print(exist)

if st.button("Insert"):

if exist ==[]:

spec=collection1.find\_one({"spec": s})

if spec is not None:

a=spec['course']

a.append(c)

collection1.update\_one({"spec": s}, {"$set":{'course':a}})

st.success(f"Data appended successfully to key: {c}")

else:

# Create a new document if key does not exist

collection1.insert\_one({'spec':s,'course':[c]})

st.success(f"New key created, data inserted: {s}")

else:

st.warning("The Subject Already Exist")

elif page == "Course View":

st.title("Course View")

st.write("Welcome to the Course View.")

documents = collection1.find({}, {"spec": 1, "\_id": 0})

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

option = st.selectbox("Specialization",(key\_values))

if option:

# Find all documents where the key exists

courses = collection1.find({'spec':option})

if documents:

st.write(f"Documents with the key '{option}':")

st.dataframe(courses) # Display documents in a table format

else:

st.write(f"No documents found with the key: {option}")

elif page == "Course Assign":

st.title("Course Registration")

st.write("Welcome to the Course page.")

documents = collection1.find({}, {"spec": 1, "\_id": 0})

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

s = st.selectbox("Specialization",(key\_values))

documents = collection1.find({"spec":s}, {"course": 1, "\_id": 0})

key\_values = [doc['course'] for doc in documents if 'course' in doc]

c = st.selectbox("course",key\_values[0])

docum = collection.find({"role":"Instructor","spec":s}, {"id": 1, "\_id": 0})

key\_values = [doc['id'] for doc in docum if 'id' in doc]

i = st.selectbox("Instructor",(key\_values))

if st.button("Insert"):

spec=collection2.find\_one({"spec": s,'course':c})

if spec is not None:

# Append new values to the existing array

a=spec['Instructor']

a.append(i)

collection2.update\_one({"spec": s,'course':c}, {"$set":{'Instructor':a}})

st.success(f"Data appended successfully to key: {c}")

else:

# Create a new document if key does not exist

collection2.insert\_one({'spec':s,'course':c,'Instructor':[i]})

st.success(f"New key created, data inserted: {s}")

elif page == "Assign View":

st.title("Assign View")

st.write("Welcome to the Assign View.")

documents = collection1.find({}, {"spec": 1, "\_id": 0})

if documents!=[]:

key\_values = [doc['spec'] for doc in documents if 'spec' in doc]

option = st.selectbox("Specialization",(key\_values))

if option:

# Find all documents where the key exists

courses = collection2.find({'spec':option})

if documents:

st.write(f"Documents with the key '{option}':")

st.dataframe(courses) # Display documents in a table format

else:

st.write(f"No documents found with the key: {option}")

elif page == "Notification":

st.session\_state["messages"]=[i for i in sum.find({})][0]

st.title("Customer Service")

m=[]

c=cust.find({},{'query':1,'id':1,'\_id':0})

if c !=[]:

for j in c:

a=f'{j["id"]}.{j["query"]}'

m.append(a)

option = st.selectbox("Notifications",(m))

if option is not None:

o=option.split('.')

for msg in st.session\_state.messages[o[0]]:

st.chat\_message(msg["role"]).write(msg["content"])

if prompt := st.chat\_input():

st.session\_state["messages"][o[0]].append({"role": "admin", "content": prompt})

st.chat\_message("admin").write(prompt)

if st.button('clear'):

cust.delete\_one({'id':o[0],'query':o[1]})

st.session\_state.admin\_joined[st.session\_state['userid']]=False

sum.delete\_many({})

sum.insert\_one(st.session\_state["messages"])

else:

st.write('No Notifications')

if st.button("Logout"):

logout()

# Control access

if not st.session\_state["reg\_in"]:

if not st.session\_state["logged\_in"]:

login()

else:

if st.session\_state['role']=='Student':

mains()

if st.session\_state['role']=='Admin':

maina()

if st.session\_state['role']=='Instructor':

maini()

else:

reg()